視覚化システムの構築と利用

関西学院大学理工学部
情報科学科 西谷研究室 4663 戸賀瀬健介

平成 21 年 2 月 21 日

概 要

物理学の理論，数式を実際の現象において考えるには，そのモデル化が重要になるが，模式的なモデルではその理解が困難となる．西谷研究室で主に取り扱っている物理現象として，析出現象の自由エネルギー計算，SiC の液相成長や SiC, SiO_2 の有限温度における高温安定性などが挙げられる．析出現象を理解するにあたり，まず初期状態からどのような過程を経て，終状態となるのかといった動力学として考える必要がある．この入口でより明確なイメージを持つことは初学者への教授や研究の引き継ぎを容易にする．また，液相成長を理解するにはナノサイズにおける結晶成長過程（原子の拡散，吸着，結晶への取込）等をイメージする必要がある．さらには，SiC や SiO_2 が温度や圧力に応じた構造の変化をイメージすることができれば研究の助けとなる．本研究では，物理現象を 3DCG によって視覚化させることで，より明確なイメージを与え，直感的に理解できるようにする目的で行った．

物理現象を視覚化させるツールとして，3DCG ソフトウェアである Maya を使用した．Maya は映画，ゲーム，CM の制作にも使用されるプロ仕様のハイエンドソフトであり，非常に高度な機能を有している．その一つに MEL というスクリプト言語がある．Maya はその GUI の全てを MEL で実行することが可能となっている．結晶の格子モデルや物理現象を再現するにあたり，その理論，数式に基づいた数値計算を行う必要があるが，MEL は Maya 独自のスクリプト言語であり，MEL を利用する環境や構文が数値計算に向いていない．そこで本研究では，MEL より計算環境の優れた Ruby によって，数値計算を行い，その結果を反映させた MEL スクリプトを自動生成させ，最終的に Maya で MEL スクリプトを読み込むことで，視覚化を実現させた．また，この一連の作業を効率よく行えるシステムを構築した．

本研究では，西谷研究室で行われている第一原理計算によるエネルギー計算を補うため，第一原理計算ソフト VASP で用いられる原子モデルの入出力ファイルからの結晶の構造や表面の視覚化を行った．さらに，肉眼では可視できないナノサイズにおける原子の挙動を再現した．具体的な成果として，格子モデルの視覚化は第三章に，物理現象のモデルは第四章にそれぞれ記述する．

目 次

第1章　緒言

物理学は，自然界の現象とその性質を物質とその間に働く相互作用によって理解したり，物質をより基本的な要素に還元して理解する自然科学の一分野である．自然界の現象とその性質を理解するためには，難解な物理学の理論，数式を学ばなければならない．しかし，初学者にとって物理学の理論，数式を理解するのは困難である．そこで，理論や数式が自然界のどのような現象を解するのかを直感的に理解するために，物理現象のモデル化を行う必要がある．
西谷研究室では，研究活動に原子モデル構築ソフト MedeA や数値計算ソフト Maple 等を用いている．しかし，これらのソフトは視覚表現に乏しい上，システムとして完成されており，自由度に欠ける．研究に対しあらゆる角度からアプローチするためにも，それに対応できる柔軟なシステムが求められている．

1.1　結晶構造

我々の日常の中で結晶は身近な存在である．水晶，ダイアモンド等の天然鉱物，雪や氷，卓上にある食塩，化学調味料，コンピュータやテレビ，オーディオ機器の中の IC 回路に使われている人工育成した半導体などが結晶にあたる．これらの全ては，それを構成する原子や分子が規則正しく周期配列してできた固体であり，我々はそれを結晶と呼んでいる [1].

SiC:　シリコンカーバイド (SiC) はその性質から次世代半導体パワーデバイスとして注目されおり，各所で積極的に研究が進められている．関西学院大学においても理工学部の金子教授らによって SiC の新奇な結晶成長プロセスである準安定溶媒エピタキシー (Metasteble solvent Epitaxy：MSE) が発明されており，同大学の西谷研究室でも SiC の研究が行われている．MSE では，基板と原料板に用いる SiC の結晶多形 (ポリタイプ) の違いによって生じる化学ポテンシャルの差が駆動力となっている [2]．このように結晶構造の違いは物性を支配している一つの要素になっているため，その理解が材料研究の出発点になる．

SiO_2:　主要なシリコン (Si) 単結晶育成法であるチョコラルスキー法 (Czochralski 法：CZ 法) では，多結晶 Si を石英 (SiO_2) 坩堝で融液化させるため，融液中に酸素原子が紛れ込む．CZ 法で引き上げられた Si 単結晶には，その酸素原子が過飽和に

取り込まれることになる．これは，その後のデバイス作成工程，特性に大きな影響を与えるため，酸素の挙動の制御が求められる．デバイス中のシリコンと酸素の関わりを解明するため，凝集の極限ともいえる SiO_2 結晶についての理解が求められる [3].

格子モデル: 結晶構造を視覚化するには，原子を ball，原子間の結合を stick で表した格子モデルを作成するのが一般的である．しかし，ball や stick での表示でもなお複雑な場合，多面体による表示法もある．
その種類は表 (1.1) のようになる．

表 1.1: 格子モデル表現法一覧

ball 表示法	原子のみ表示する方法
stick 表示法	結合のみ表示する方法
stick&ball 表示法	原子と結合の両方を表示する方法
配位多面体表示法	多面体によって結晶構造の簡素化表示する方法

1.2 Fe-Cu の核生成

Fe-Cu 系は Fe 系ナノメタルで次世代の鉄鋼材料として注目されている．核生成は組織発展のもっとも初期に起こる現象であり，組織制御のシミュレーションにおいては非常に重要である．この核生成について理解するには，まず動力学的にとらえ，原子レベルでの実際の動きを考える必要がある．その動きを理解するため，モデル化したものが，図 1.1 の古典的核生成モデルである．

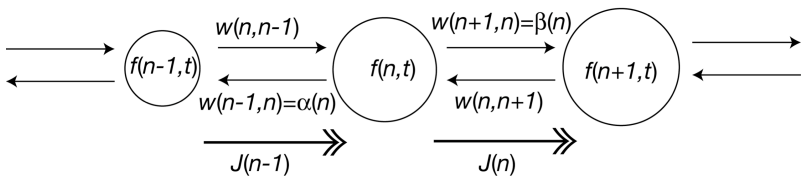

図 1.1: 古典的核生成モデル

このモデルの $f(n,t)$ は核の分布，密度を示した関数であり，これはクラスターサイズ n に依存する（t は時間）．確率 α で核が縮小し，確率 β で成長する様子

が示されている．$J(n)$ とは確率 α β の和であり，系全体でとらえると，$J(n) = 0$ となる．しかし，系を局所的にみると，確率の変動で核の成長，縮小は起こっている．それを示したのが，式 (1.1)，式 (1.2) で示される詳細つりあい条件である．

$$\alpha(n+1) = \beta(n)\frac{\nu(n)}{\nu(n+1)} \tag{1.1}$$

$$\nu(n) \propto exp(\frac{-\Delta F(n)}{kT}) \tag{1.2}$$

ν (n) はクラスターがその状態でいる確率であり，これは自由エネルギーに依存する．その自由エネルギーの変化を示したグラフが図 1.2 となる．

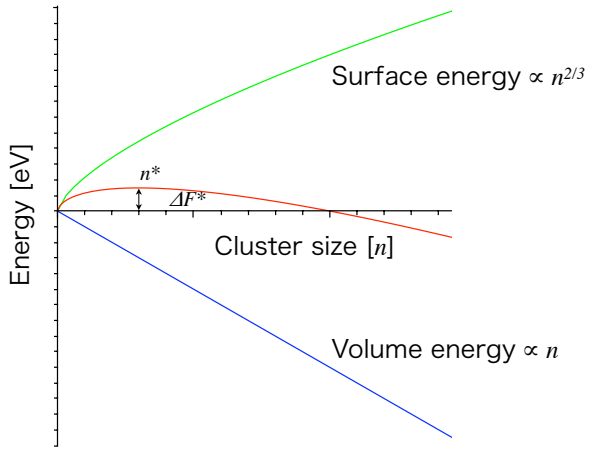

図 1.2: クラスターの自由エネルギー変化 [5]

下の青線が駆動力となる体積エネルギーで，上の緑が表面エネルギーとなっている．その差としてクラスターの自由エネルギーが得られる．これらはクラスターサイズに依存し，大きくなるほど安定する [4].
ナノスケールの物理現象の扱いには，そのモデル化が出発点となる．初学者がこのような物理現象を取り扱う際には，図 1.1 のような抽象的なモデルでなく，より具体的なモデルが求められる．

1.3　SiC 表面上における C 原子の拡散

小節 1.1 でも記したように，西谷研究室でも SiC の研究が行われている．MSE による単結晶成長機構を解明するにあたり，結晶成長の素過程となる拡散についての知見を得ることは必要である．なお，図 1.3MSE の構成上，基板となる 4H-SiC は液体 Si に覆われている．そのため，表面拡散を行う Si 原子は，液体 Si から充分に獲得することが可能であり，結晶成長を行う上で Si 原子が不足することは無いと考えられる．つまり，結晶成長は C 原子が取り込まれる速度によって律速されると考えられるため，C 原子の拡散が重要となる．

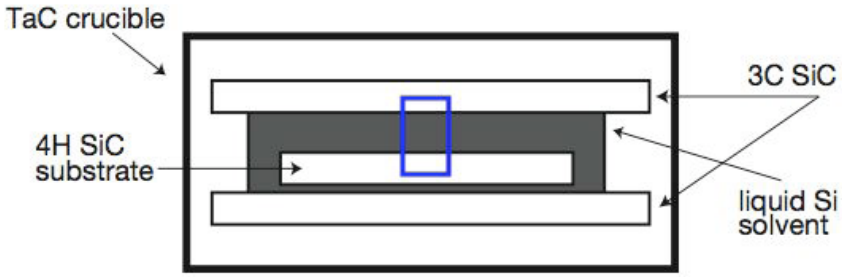

図 1.3: MSE の構成．TaC 坩堝の中で，原料板となる 3C-SiC で基板となる 4H-SiC と薄膜の液体 Si を挟み込む．これを一定温度 (1700 ℃) で放置する．3C-SiC と 4H-SiC の化学ポテンシャルの差により 3C-SiC より溶け出した C 原子は，4H-SiC に吸着し，結晶成長する．

第2章　手法

物理モデルを視覚化させるシステムとして，[Ruby - MEL - Maya] を提案する．

2.1　Maya

Maya は，オープンアーキテクチャを基板とした強力な統合型の 3D モデリング，アニメーション，レンダリングソリューションである．多くのフィルムやビデオアーティスト，ゲーム開発者，マルチメディアデザイナー，3DCG に関わる SOHO デザイナーなどが使用しているプロ仕様のハイエンドソフトである．グラフィックス性に優れ，非常に高度な機能を有しているため，自由度も高い．また，Maya はその GUI の全てを MEL というスクリプト言語で実行可能となっている．本研究では，実際に視覚化を行うインターフェースとして Maya を選定した．

MEL　MEL(Maya Embedded Lanquage) とは Maya で使用できるスクリプト言語であり，Maya の GUI の機能の全てをまかなうことが可能となっている．MEL のみで CG の作成，カスタマイズを行うことができるが，Maya 専用のスクリプトエディタ上でしか実行できない．また，線形計算などの数値計算に向いておらず，構文も複雑な構成となっている [6]．

2.2　Ruby

Ruby とはまつもとゆきひろ氏により開発されたオブジェクト指向スクリプト言語である．一般的に，Ruby は数値計算には向いていないと言われる．実際に，Ruby は C や Fortan などの他の言語と比べ，実行速度などの面で遅い．しかし，Ruby の言語は，単純な構文で書かれており，可読性に優れている．さらにコンパイルを必要としないインプリンタ方式を採用しているため，実行の手間が少ない．加えて，高機能なスクリーンエディタとして有名な Emacs と併用すれば，構文に応じてスクリプトが色分けされるので．開発環境が非常に良いものとなる．本研究では，MEL の複雑な計算環境に代わるインターフェースとして Ruby を選定した．

2.3 Ruby - MEL - Maya

視覚化システム [Ruby - MEL - Maya] の実行の流れについて解説する．先に記したように，視覚化を実現させるツールとして，Ruby，Maya を使用した．まず，ユーザが数値計算を行うメインスクリプトを Ruby にて作成，実行する．そのデータを反映させた MEL スクリプトを出力させる．そして，Maya のインターフェースで生成した MEL スクリプトを読み込むことで視覚化を実現させる．また，この作業を効率良く行うため，視覚化の目的に応じてカスタマイズを行う関数のライブラリを作成した．

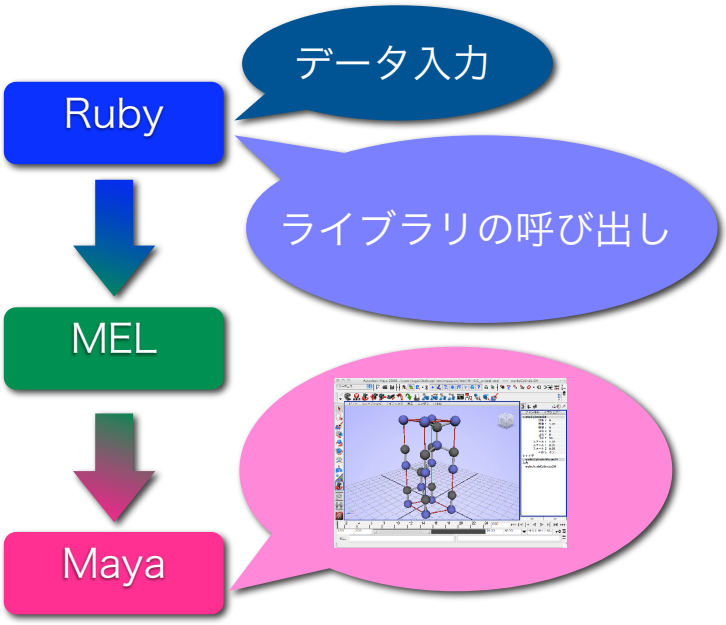

図 2.1: Ruby-MEL-Maya の流れ

第3章　視覚化のシステム

本章では，[Ruby - MEL - Maya] のシステムの実装，使用法を解説する．

3.1　データディレクトリ

格子モデルを作成するに必要な原子位置を示すデータを格納しておく．格子モデルを作成するパラメータとして，結晶の並進対称性を規格化する基本並進ベクトル（Primitive Vector）と，それを元に原子位置を示したベクトル（Basis Vector）が必要になる．Primitive Vector を式 (3.1)，Basis Vector を式 (3.2) とした場合，実空間における原子位置（X,Y,Z）は式 (3.3) で表される．

$$P_0 = \begin{bmatrix} p_{0,0} \\ p_{0,1} \\ p_{0,2} \end{bmatrix}, P_1 = \begin{bmatrix} p_{1,0} \\ p_{1,1} \\ p_{1,2} \end{bmatrix}, P_2 = \begin{bmatrix} p_{2,0} \\ p_{2,1} \\ p_{2,2} \end{bmatrix} \tag{3.1}$$

$$B = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.2}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = P_0 x + P_1 y + P_2 z \tag{3.3}$$

以下の小節にデータソースとその形式を紹介する．なお，データのサンプルは，いずれも 4H-SiC としている．基本的にデータをコピーしたファイルを作成し，それを入力ファイルとして，メインスクリプトに読み込ませる．スペース，改行，データの位置，その他キーワードは構文解析時に意味を持つので，データファイルへの追記または削除には注意を要する．

3.1.1 web データ

データソースは結晶構造のデータを公開している 'Crystal Lattice Structures' というwebページである [8].
また，構文解析時のキーワードは'Primitive'，'Basis' となっている.

web データ

```
4H-SiC

Primitive vectors
a(1) =   1.54025500 -2.66779992  0.00000000
a(2) =   1.54025500  2.66779992  0.00000000
a(3) =   0.00000000  0.00000000 10.08480000


Volume =  82.87874525


Reciprocal vectors
b(1) =   0.32462157 -0.18742035  0.00000000
b(2) =   0.32462157  0.18742035  0.00000000
b(3) =   0.00000000  0.00000000  0.09915913


Basis Vectors:
Atom    Lattice Coordinates               Cartesian Coordinates

Si  0.00000000  0.00000000  0.00000000     0.00000000  0.00000000  0.00000000
Si  0.00000000  0.00000000  0.50000000     0.00000000  0.00000000  5.04240000
C   0.00000000  0.00000000  0.18750000     0.00000000  0.00000000  1.89090000
C   0.00000000  0.00000000  0.68750000     0.00000000  0.00000000  6.93330000
Si  0.33333333  0.66666667  0.24982500     1.54025500  0.88926664  2.51943516
Si  0.66666667  0.33333333  0.74982500     1.54025500 -0.88926664  7.56183516
C   0.33333333  0.66666667  0.43732500     1.54025500  0.88926664  4.41033516
C   0.66666667  0.33333333  0.93732500     1.54025500 -0.88926664  9.45273516
```

3.1.2 POSCAR

データソースは第一原理計算ソフト VASP の原子位置入力ファイルとなる POSCAR である．
また，構文解析時のキーワードは’Direct’ となっている．

POSCAR

```
4H-SiC
    1.000000000000000
     3.0800000000000000    0.0000000000000000    0.0000000000000000
    -1.5399999300000000    2.6673582800000000    0.0000000000000000
     0.0000001800000000    0.0000003200000000   10.0810000000000000
   4   4
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.0000000000000000  0.0000000000000000  0.5000000000000000
  0.3333333300000021  0.6666666699999979  0.2500000000000000
  0.6666666699999979  0.3333333300000021  0.7500000000000000
  0.0000000000000000  0.0000000000000000  0.1875000000000000
  0.0000000000000000  0.0000000000000000  0.6875000000000000
  0.3333333300000021  0.6666666699999979  0.4375000000000000
  0.6666666699999979  0.3333333300000021  0.9375000000000000
```

3.1.3 OUTCAR

データソースは第一原理計算ソフト VASP の出力ファイルとなる OUTCAR の一部である．
補足として，原子数のデータ [atoms: 4 4] を付け足した．
また，構文解析時のキーワードは’atoms:’，’POSITION’ となっている．
なお，このファイルには Primitive Vector が無いため，結晶の単位格子を特定出来ない．そのためセルの拡張を行うことが出来ないファイルとなっている．

OUTCAR

```
4H-SiC
atoms: 4 4
 POSITION                                       TOTAL-FORCE (eV/Angst)
 -----------------------------------------------------------------------------------
      0.00000      0.00000      0.00000         0.000000      0.000000     -0.074116
      0.00000      0.00000      5.04050         0.000000      0.000000     -0.074116
      0.00000      1.77824      2.52025         0.000000      0.000000     -0.044730
      1.54000      0.88912      7.56075         0.000000      0.000000     -0.044730
      0.00000      0.00000      1.89019         0.000000      0.000000     -0.036570
      0.00000      0.00000      6.93069         0.000000      0.000000     -0.036570
      0.00000      1.77824      4.41044         0.000000      0.000000      0.155415
      1.54000      0.88912      9.45094         0.000000      0.000000      0.155415
 -----------------------------------------------------------------------------------
    total drift:                                -0.000012      0.000014      0.002052
```

3.2 ライブラリ

視覚化させる格子モデルや物理現象をカスタマイズする上で，用途に応じた構文を関数として作成し，ライブラリとして格納した．その種類として，データ入力を行う関数，格子モデルの拡張や作成を行う関数，オブジェクトの作成，オブジェクトの移動，スケールの変更，回転を行う関数などがある．

input: input には，input 関数，input_pos 関数，input_out 関数がある．それぞれ小節で記した web データ，POSCAR，OUTCAT 用の入力関数となっている．使用法は入力ファイル名と入力ファイルのある場所を引数とし，実空間上の原子位置ベクトル，基本並進ベクトルと原子の種類を返す．なお input_out のみ，基本並進ベクトルのデータが存在しないので，原子位置ベクトルと原子の種類を返す．

Ruby の記述例

```
data = '4H-SiC.txt'
path = 'data/POSCAR'
ball,p_vec,name = input_pos(data,path)

# ball は実空間上の原子位置ベクトルを格納している．
# なお，ball は原子の種類名をキーとしたハッシュ関数となっている
# ハッシュキーの要素中に原子位置ベクトルを格納した配列がある．

# p_vec は基本並進ベクトルとなる 3 つのベクトルを格納している．

# name は原子の種類を格納している．なお POSCAR，OUTCAR では原子の種類が特定できないため，
# 自動で atom0，atom1..atomN と順に名前をつける．
```

supercell supercell には，格子モデルの拡張を行う supercell 関数がある．実空間上の原子位置ベクトル，基本並進ベクトル，拡張サイズを引数とし，拡張させた原子位置ベクトルを返す．

Ruby の記述例

```
newball = Hash.new
cell_scale = [[-1,1],[-1,1],[-1,1]]
name.each do |key|
  newball[key] = supercell(ball[key],p_vec,cell_scale)
end
```

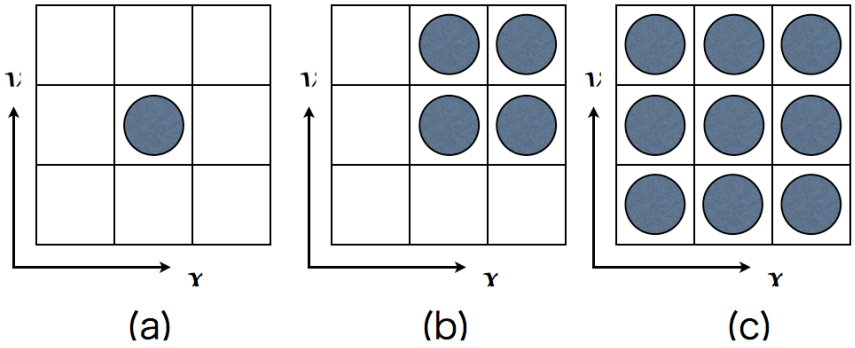

図 3.1: supercell の拡張方法：拡張サイズの引数の形式として cell_scale=[[x0,x1],[y0,y1],[z0,z1]] とする．この x, y, z は，それぞれ input の返り値 p_vec より p_vec[0], p_vec[1], p_vec[2] 方向の拡張サイズとなる．(a) は拡張を行っていない状態である．cell_scale=[[0,1],[0,1],[0,0]] とすると (b) のように拡張される．cell_scale=[[1,1],[1,1],[0,0]] とすると (c) のように拡張させる．

color: color には，色を作成する color_lambert 関数，color_blinn 関数と色をカスタマイズする color_transparency 関数，color_incandescence 関数がある．lambrt は最もよく使用される関数である．ライトが均一に拡散されるため光沢やつやのない材質を表現できる．blinn は lambert と違い，オブジェクト表面に光沢やつやが出来るため金属などの材質に用いるとよい [7]．transparency は色を透明にし，incandescence は色を光らせる．color_lambert 関数，color_blinn 関数は色を指定する RGB 値を引数とし，色を設定する MEL スクリプトを返す．color_transparency 関数，color_incandescence 関数は色の変数と RGB 値を引数とし，色をカスタマイズする MEL スクリプトを返す．

Ruby の記述例

```
color = []
file1,color[0] = color_lambert(file1,[1,0,0])

file1,color[1] = color_lambert(file1,[0,1,0])
file1 = color_transparency(file1,color[1],[0.5,0.5,0.5])

file1,color[2] = color_lambert(file1,[0,0,1])
file1 = color_incandescence(file1,color[2],[0.5,0.5,0.5])

# file1 は MEL スクリプトを格納しているファイル変数である.
```

make_cell: make_cell には，格子モデル作成する make_ball_cell 関数と make_stick_cell 関数がある．これは小節 1.1 で記した ball 表示法と stick 表示法になる．また stick&ball 表示法は，この二つを併用すれば可能となる．make_ball_cell 関数は実空間上の原子位置ベクトル配列，原子の種類名，色の変数と ball の scale 値を引数とし，ball の格子モデルを作成する MEL スクリプトを返す．make_stick_cell 関数は結合させる二種類の原子位置ベクトル配列，ボンドの長さの上限下限，色の変数と stick の scale 値を引数とし，ball の格子モデルを作成する MEL スクリプトを返す．なお，一種類しかない格子モデルの場合，同一原子位置ベクトル配列を二つ引数とすれば良い．scale 値については長さは原子間距離がデフォルトとなっているので，変化するのは太さの数値となっている．

Ruby の記述例

```
  name.size.times do |i|
    key = name[i]
    c = color[i+1]
    scale = [0.5,0.5,0.5] # ball の scale 値
    file1 = make_ball_cell(file1,ball[key],key,c,scale)
  end

  scale = [1,0.1,0.1] # stick の scale 値 1=stick の長さ，0.1=stick の太さに相当する.
  len = [0,2] # 0 <= 原子間距離 < 2 のとき，stick を作成する.
  file1 = make_stick_cell(file1,ball1['atom0'],ball1['atom1'],len,color[3],scale)

# file1 は MEL スクリプトを格納しているファイル変数である.
# color[i] は色を指定する MEL スクリプトを格納している配列である.
# この場合，atom0 の色が color[1]，atom1 の色が color[2]，stick の色が color[3] となる.
```

cell_support: cell_support には，格子モデルに螺旋転位を施す make_spiral 関数と格子モデルのユニットセルをワイヤーフレームで囲う wireframe_unitcell 関数がある．make_spiral 関数は原子位置ベクトル配列，バーガースベクトルとコアの半径を引数とし，螺旋転位を施した原子位置ベクトル配列を返す．wireframe_unitcell 関数は基本並進ベクトル配列と色の変数を引数とし，ユニットセルを囲うワイヤーフレームを作成する MEL スクリプトを返す．

Ruby の記述例

```
name.each do |key|
  sp_ball = make_spiral(ball[key],burgers,core)
end

file1 = wireframe_unitcell(file1,p_vec,color[0])

# 転位は Maya における y 軸方向に起こすので，burgers はズラす距離の値を入れる．
# core は螺旋転位の中心にできるコア（穴）の半径の値を入れる．
```

object: object には，オブジェクトを作成する object_add 関数がある．object_add 関数はオブジェクトの種類，オブジェクトにつける任意の名前と色の変数を引数とし，任意のオブジェクトを作成する MEL スクリプトを返す．なお，作成されたオブジェクトは位置 [0,0,0]，スケール [1,1,1]，回転 [0,0,0] がデフォルトとなっているので，後に記してある curretTime ライブラリと併用するのが良い．

Ruby の記述例

```
name = 'add_atom'
file1 = object_add(file1,'sphere',name,color[0])
```

表 3.1: オブジェクト一覧

種類の入力名	作成されるオブジェクト
sphere	球
cylinder	円柱
cone	円錐
nurbsCube	長方体

currentTime: currentTime には，オブジェクトを移動させる curretTime_move 関数，オブジェクトのスケールを変更する curretTime_scale 関数，オブジェクトを回転させる curretTime_rotate 関数　色の変更する curretTime_color 関数　色の透明度を変更 curretTime_color_trans 関数と色の輝度を変更する curretTime_color_incan 関数がある．これらの関数は，アニメーションを作成する時に必要となるキーフレーム値，オブジェクトの名前もしくは色の変数と変更値を引数として，オブジェクトを変更する MEL スクリプトを返す．アニメーションを作成する必要がない場合，キーフレーム値は'0' にしておくと良い．

Ruby の記述例

```
file1 = currentTime_move(file1,0,obj_name,[1,1,1])
file1 = currentTime_scale(file1,0,obj_name,[2,2,2])
file1 = currentTime_rotate(file1,0,obj_name,[0,90,0])

currentTime_color(file1,0,color[0],[0,1,0])
currentTime_color_trans(file1,0,color[0],[1,1,1])
currentTime_color_incan(file1,0,color[0],[1,1,1])
```

例えば，ボールを [0,0,0] の位置から [10,10,10] の位置まで移動させるアニメーションを作成する場合，以下のように記述する．

Ruby の記述例

```
file1 = object_add(file1,'sphere','ball',color[0])

file1 = currentTime_move(file1,1,'ball',[0,0,0])

file1 = currentTime_move(file1,50,'ball',[10,10,10])
```

このように記述すると，ボールが [0,0,0] から [10,10,10] まで５０フレームかけて移動する．

light: light には，ライトを作成する ambientLight 関数，directionalLight 関数と spotLight 関数がある．ライトの関数は，それぞれライトにつける任意の名前，光の色と光度を引数とし，ライトを作成する MEL スクリプトを返す．なお，spotLight には，コーンアングルを指定する引数がある．ライトの移動，スケール変更や回転は curretTime で変更できる．
ライトについての詳しい記述は [6] を参照．

Ruby の記述例

```
# file1 名前 光の色 光の強さ
file1 = ambientLight(file1,'light1',[1,1,1],10)

# file1 名前 光の色 光の強さ
file1 = directionalLight(file1,'light2',[1,1,1],10)

# file1 名前 コーンアングル 光の色 光の強さ
file1 = spotLight(file1,'light3',30,[1,1,1],10)
```

camera: camera には，カメラを作成する camera 関数とカメラとカメラの視点を作成する camera_aim 関数がある．camera 関数はカメラにつける任意の名前とカメラの位置を引数とし，カメラを作成する MEL スクリプトを返す．camera_aim 関数はカメラ，視点につける任意の名前とカメラ，視点の位置を引数とし，カメラと視点を作成する MEL スクリプトを返す．カメラの移動，スケール変更や回転は curretTime で変更できる．なお，camera_aim 関数を用いた場合，視点に合わせてカメラが自動で回転する．

Ruby の記述例

```
name = 'camera'
file1 = camera(file1,name,[10,0,0])

name1 = 'camera'
name2 = 'aim'
file1 = camera_aim(file1,name1,name2,[10,0,0],[0,0,0])

# [10,0,0] はカメラの位置, [0,0,0] は視点の位置となる.
```

tetra: tetra には，格子モデルの４配位の原子に対し四面体を構築する関数がある．四面体を作成する際，注目する原子を格子モデルの全原子を対象とした tetra_all 関数と任意に設定する tetra_selectc 関数がある．tetra_all 関数はキーフレーム値，四面体の重心となる原子の配列，全原子の配列，重心原子の種類，ボンドの長さと色の変数を引数とし，四面体を構築する MEL スクリプトを返す．tetra_select 関数は全原子の配列でなく，任意の原子配列を引数とする．

Ruby の記述例

```
file1 = tetra_all(file1,0,ball['atom0'],ball,'atom0',[0,2],color[0])

file1 = tetra_select(file1,0,ball['atom0'],ball['atom1'],'atom0',[0,2],color[1])

# ball は実空間上の原子位置ベクトルを格納したハッシュ
```

作成したライブラリを表 (3.1) にまとめた.

表 3.2: ライブラリ一覧

input	データファイルの入力を行う.
supercell	格子モデルの拡張を行う
color	色を作成，カスタマイズする
make_cell	格子モデルを作成する
cell_support	格子モデルの表現を補う（例：ユニットセルの枠を作成する）
object	オブジェクトを作成する
currentTime	オブジェクトの移動，スケールの変更，回転を行う
light	ライトを作成する
camera	カメラを作成する
tetra	格子モデルを四面体を用いて表現する

3.3　メインスクリプト

実際にユーザが作成する実行ファイルである．ユーザはこのメインスクリプト上で数値計算や物理モデルの作成を行う．

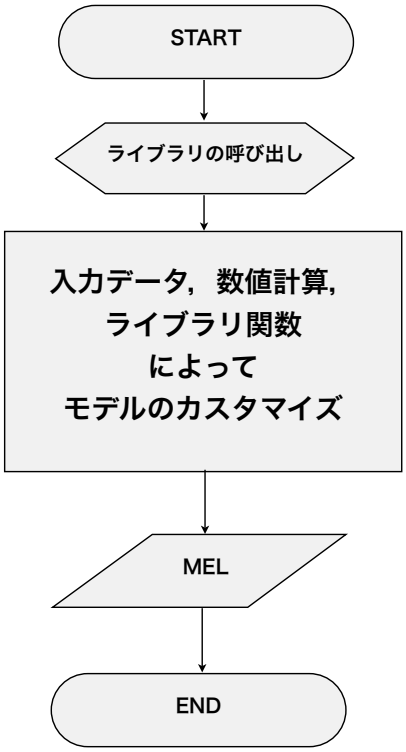

図 3.2: メインスクリプトのフローチャート

3.4 MELの出力

メインスクリプトからの出力ファイルとなる．以下に Maya への出力方法を記述する．

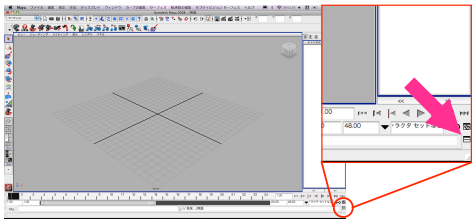

図 3.3: Maya インターフェース

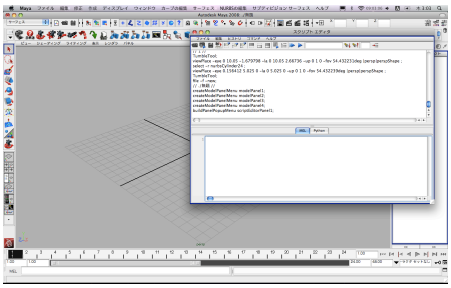

図 3.4: Maya のスクリプトエディタ

MEL スクリプトの出力には，まず Maya のスクリプト開く必要がある．図 3.1 の矢印の箇所をクリックすると図 3.2 にある Maya のスクリプトエディタが表示される．

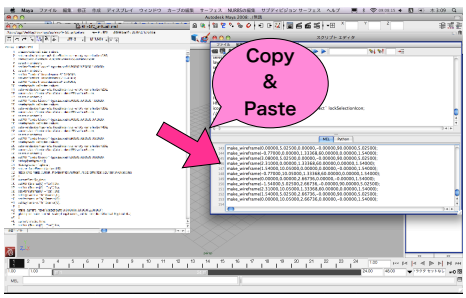

図 3.5: MEL の Copy&Paste

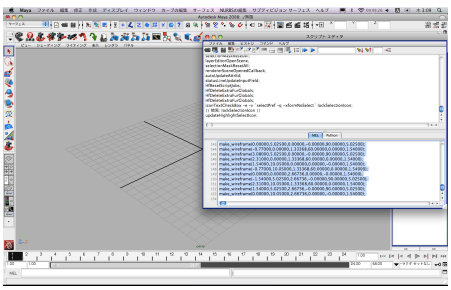

図 3.6: MEL の実行

図 3.3 のように ’mi ’などのテキストエディタで開いた MEL スクリプトをスクリプトエディタに貼付ける．MEL スクリプトを図 3.4 のように全選択した状態で，キーボードより ’Control + enter ’を入力する．

最終的に図 3.5，図 3.5 のような出力が得られる．オブジェクトの線表示とカラー表示はキーボードの ’５ ’のキーで変更が可能となっている．

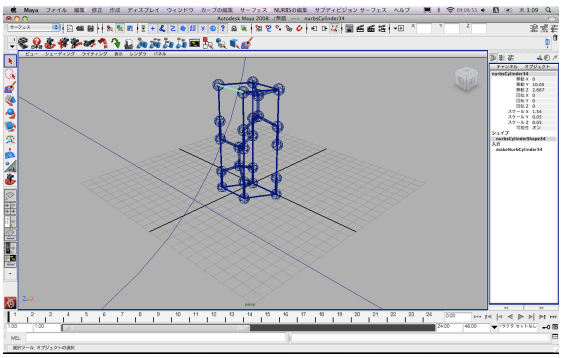

図 3.7: 出力結果（線表示）

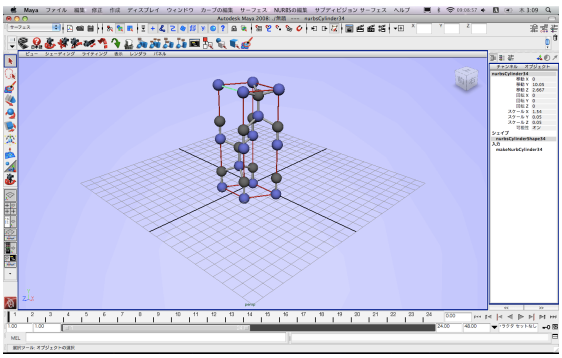

図 3.8: 出力結果（カラー表示）

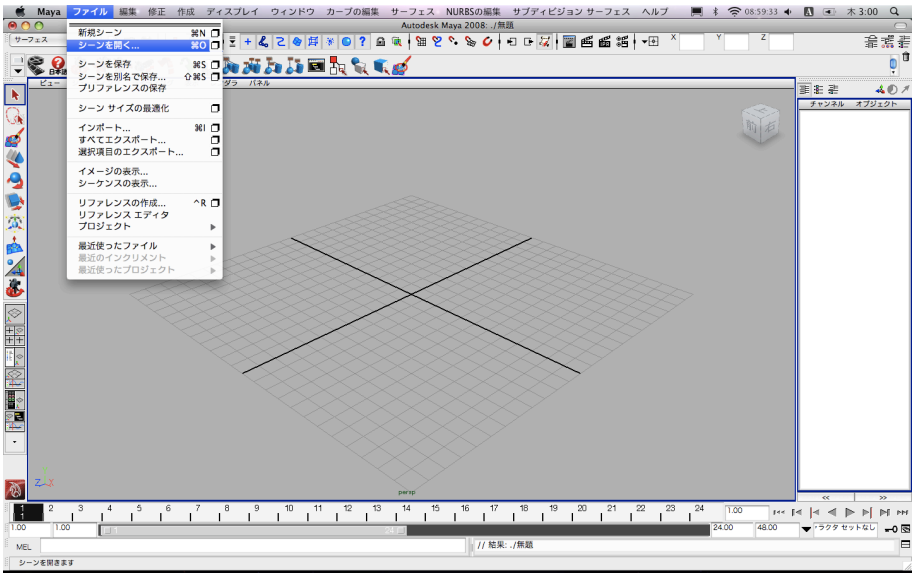

図 3.9: 簡易出力方法

また，図 3.7 の方法や ’Command + o ’で表示させる方法もある．しかし，この方法で出力した場合，画像 or 動画を作成するレンダリングの作業が失敗することがある．そのため，この方法は中間出力の確認に用いると良い．

第4章　格子モデルの視覚化

4.1　SiC 多形の視覚化

SiC には，100 種類以上の結晶多形が存在する．多くの結晶多形の中でも発生率が高く，応用上重要となる Ramsdell notation で 3C-, 4H-, 6H-SiC がある．これら結晶多形の表記法は，積層方向の単位格子に含まれる Si-C 単位層の数と，結晶系の頭文字 (C：立方晶，H：六方晶) を組み合わせて表されている．

4.1.1　SiC 結晶格子

3C-, 4H-, 6H-SiC の結晶格子を，それぞれ図 4.1，図 4.2，図 4.3 に表した．青いボールが Si 原子，黒いボールが C 原子となっている．同じ SiC にも関わらず，晶形違いと中の原子配置の様子が違う．3C-SiC は晶形が立方晶となっており，その中の原子数は Si 原子と C 原子がそれぞれ 4 個ずつ晶形中に存在している．4H-, 6H-SiC は六方晶となっており，4H-SiC は原子数 Si, C 原子 12 個ずつで構成されており，6H-SiC は原子数 Si, C 原子 18 個ずつで構成されている．

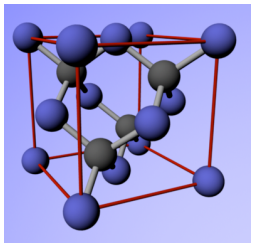

図 4.1: 3C-SiC

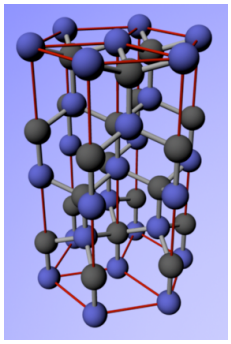

図 4.2: 4H-SiC

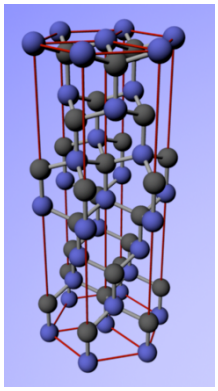

図 4.3: 6H-SiC

4.1.2 SiC 表面構造

六方晶，立方晶の代表的な面として図 4.4 がある．この図は晶形に対し面を区切っているだけで，晶形の中で原子がどのように並んでいるのか解らない．また，図 4.5 は 3C-, 4H-, 6H-SiC の結晶構造を {11-20} 面から見た模式的な断面図である．なお，左から 3C-, 6H-, 4H-SiC となっている．3C-SiC は下から順番に A,B,C と 3 周期となっている．4H-SiC は下から順番に A,B,A,C と 4 周期となっている．6H-SiC は下から順番に A,B,C,A,C,B と 6 周期となっている．

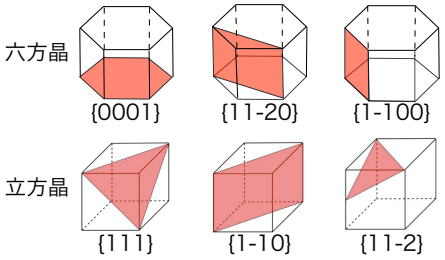

図 4.4: 六方晶，立方晶の代表的な面を示した模式図

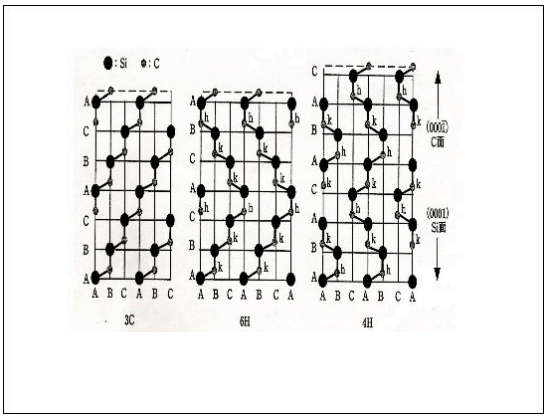

図 4.5: 3C-, 4H-, 6H-SiC の結晶構造を {11-20} 面からみた断面図

図 4.6，図 4.7 と図 4.8 は Maya によって作成した SiC 多形の表面構造である．図 4.1.2 の {111} 面上の赤いフレームに注目すれば，立方晶の {111} 面，{1-10} 面，{11-2} 面と六方晶の {0001} 面，{11-20} 面，{1-100} 面がそれぞれ等価な面である様子が確認できる．{11-2} 面に注目して見れば，図 4.5 の模式図に比べ，ボールとシリンダーが浮き出ており，構造の周期性をより解りやすく表現できた．さらに，三面との間に出来るステップの様子も表現できた．面を透明にすることで結晶構造と面を同時に確認することの可能にした．

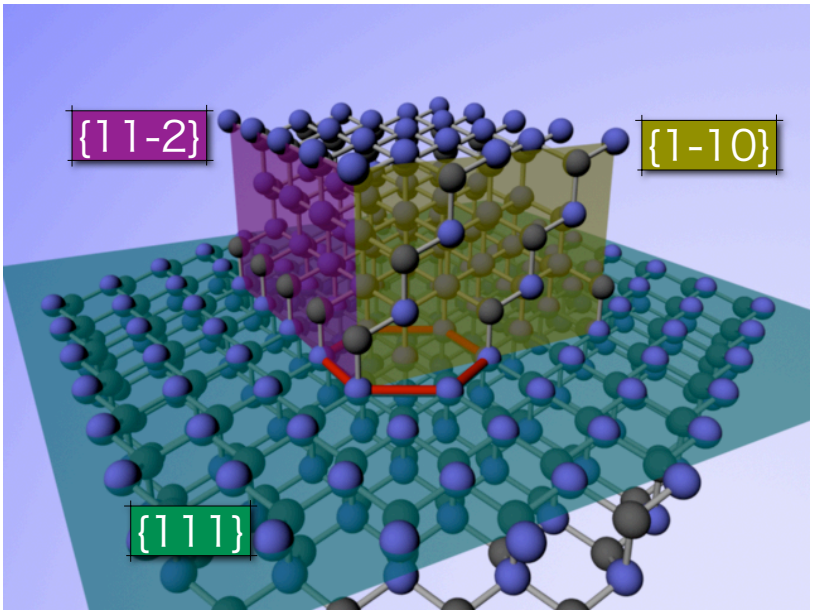

図 4.6: 3C-SiC 代表的な面

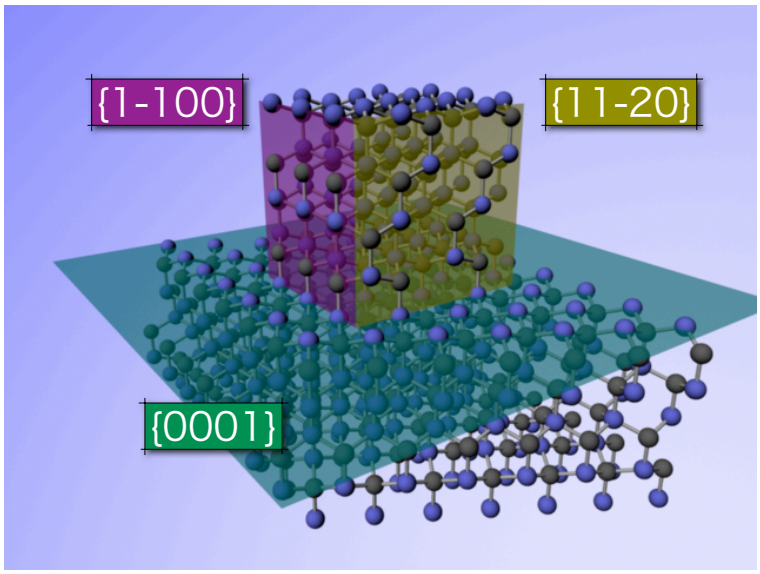

図 4.7: 4H-SiC 代表的な面

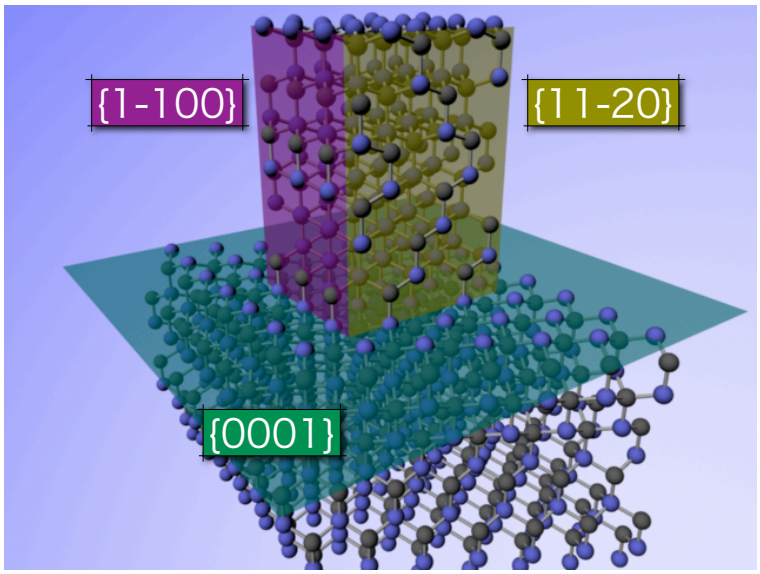

図 4.8: 6H-SiC 代表的な面

4.2 SiO_2 多形の視覚化

シリカ鉱物 (SiO_2) は，温度，圧力に応じて様々な結晶構造をとることが知られている．これらは一度できてしまうと，常温常圧でも安定に存在する．これらの SiO_2 多形は，ステショバイトを除いた全ての構造は SiO_4 四面体のフレームワークからできている．この局所的な相似性にも関わらず，密度では約２倍，体積弾性率では約１８倍もの広範囲にわたるのが SiO_2 の特徴である．図 4.9 は SiO_2 多形の P-T 状態図となっている．同図の石英 (quartz) に加え，トリディマイト (tridymite) とクリストバライト (cristobalite) にも高温型と低温型の構造を持つ．

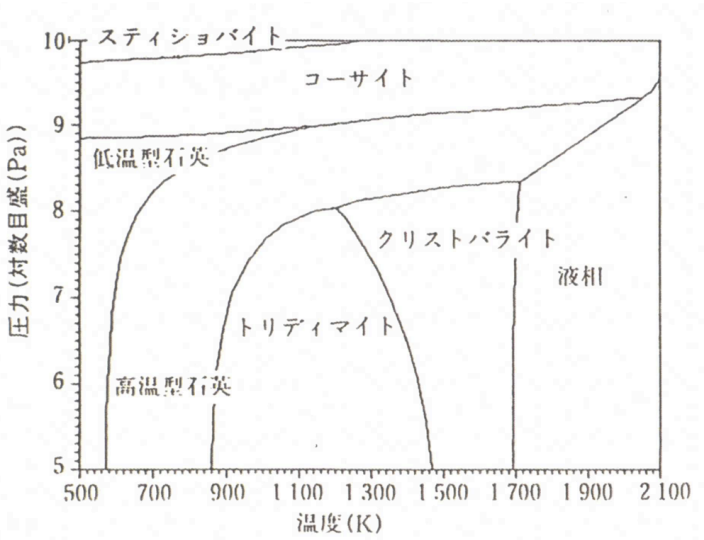

図 4.9: SiO_2 の P-T 状態図

ステショバイト (stishovite)(図 4.10) を除く全ての SiO_2 結晶構造は，図 4.11 の通りに SiO_4 ユニットで表すことが可能である．コーサイト以下，SiO_2 多形の結晶構造を stick&ball と SiO_4 四面体にてを図 4.12 から図 4.25 に示した．なお，青が Si 原子，緑が O 原子となっている．SiO_2 多形は Si-O-Si ボンドで折れ曲がっており，結晶構造が複雑化している．四面体表示を用いることで，構造全体を見やすくできた．なお，ステショバイトの構造が他の SiO_2 多形と大きく異なっているのは図 4.2.1 に示されている通り，非常に高圧な環境で安定相をとっているからと考えられる．

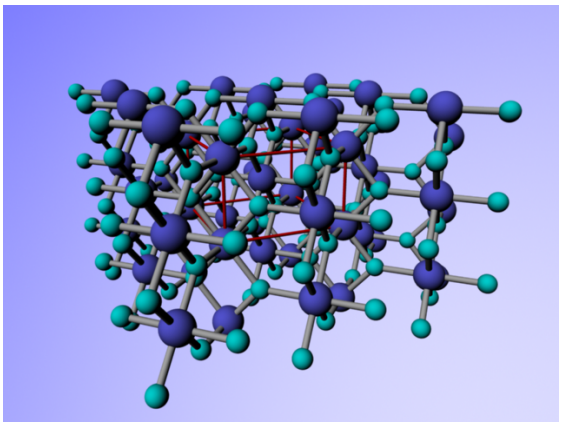

図 4.10: stishovite

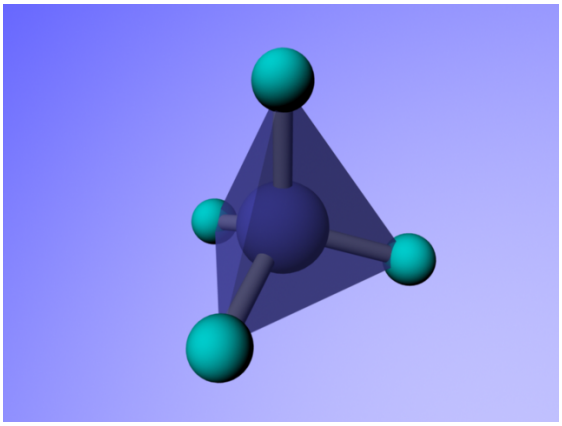

図 4.11: SiO_4 ユニット

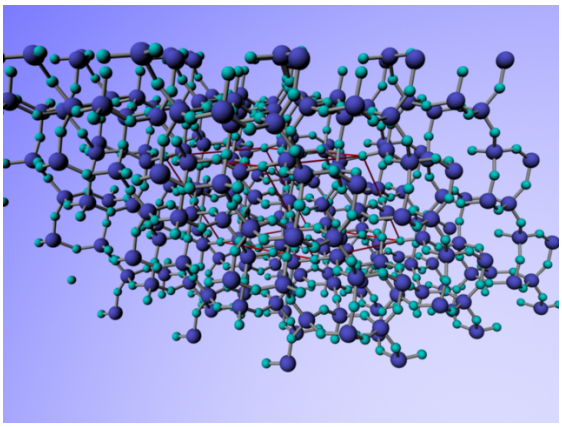

図 4.12: coesite

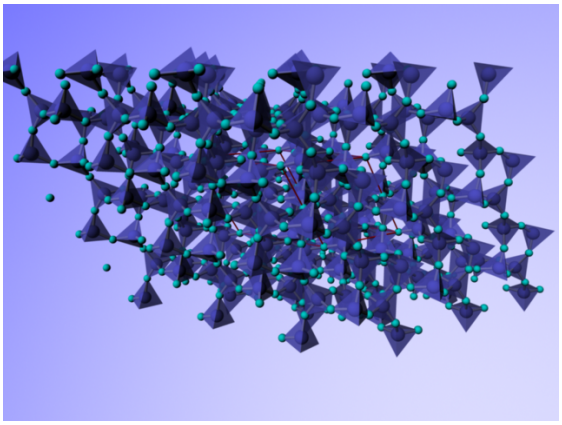

図 4.13: coesite(四面体)

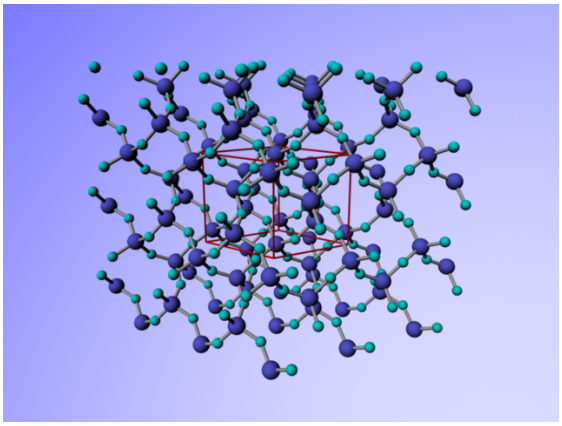

図 4.14: 低温型 quartz

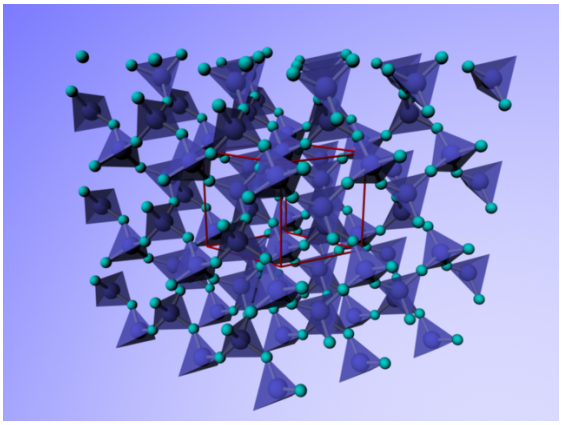

図 4.15: 低温型 quartz(四面体)

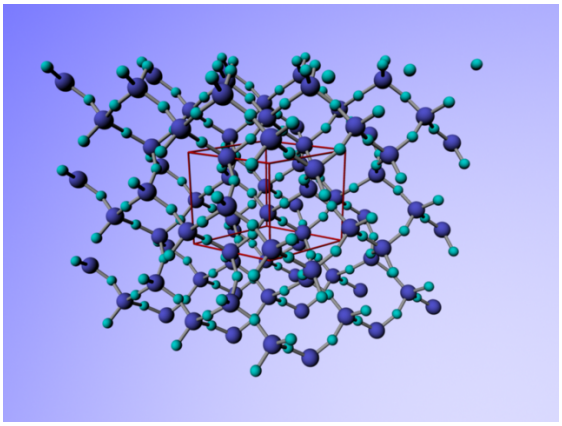

図 4.16: 高温型 quartz

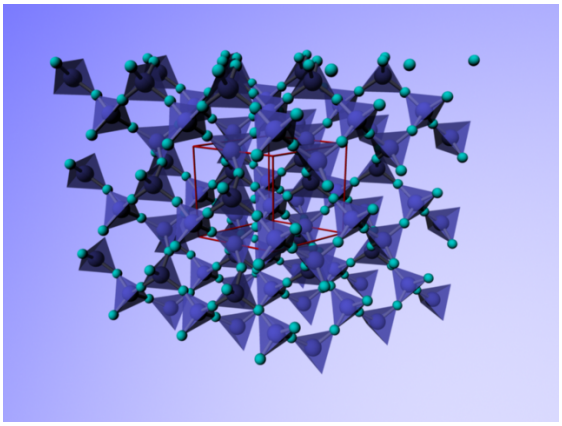

図 4.17: 高温型 quartz(四面体)

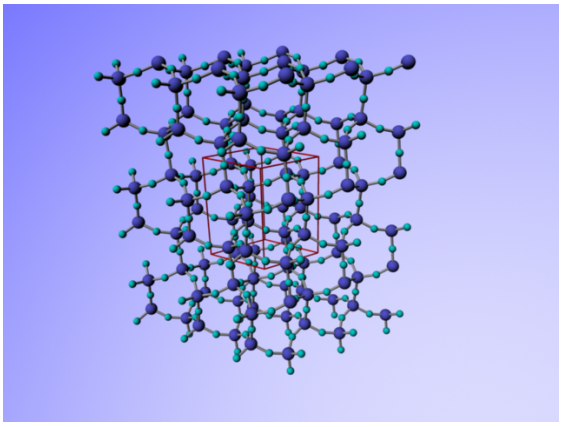

図 4.18: 低温型 tridymite

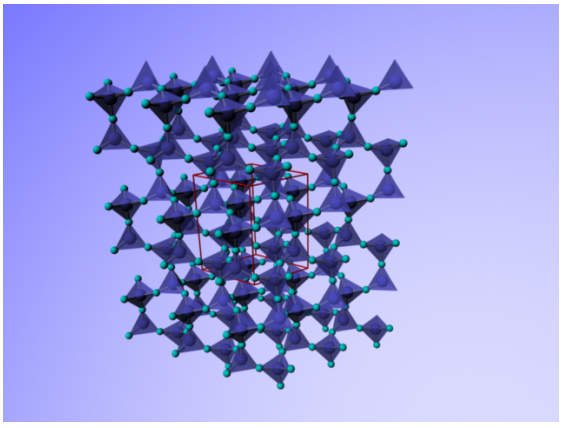

図 4.19: 低温型 tridymite(四面体)

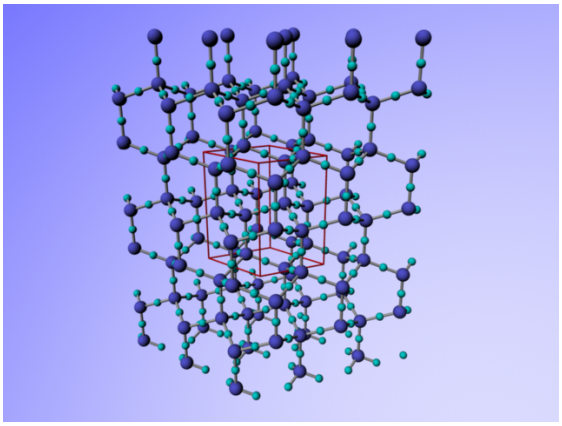

図 4.20: 高温型 tridymite

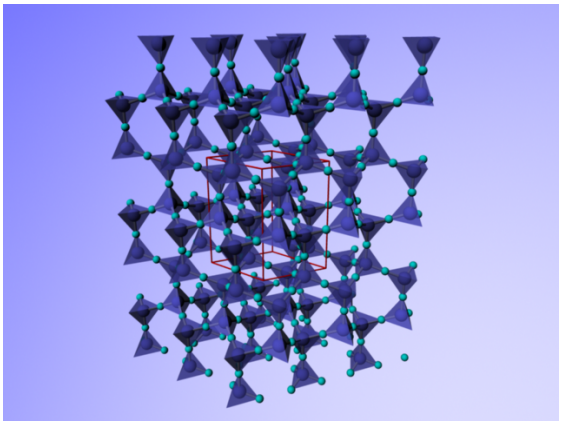

図 4.21: 高温型 tridymite(四面体)

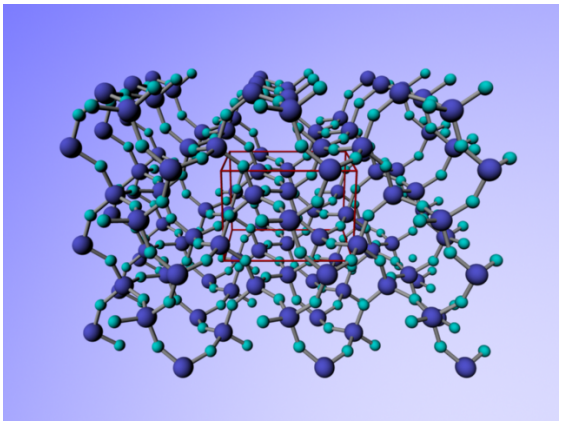

図 4.22: 低温型 cristobalite

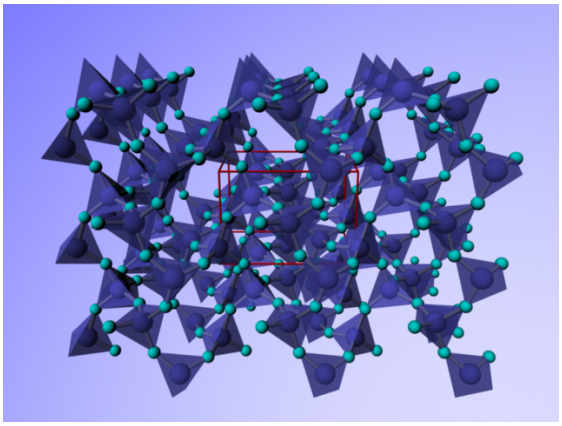

図 4.23: 低温型 cristobalite(四面体)

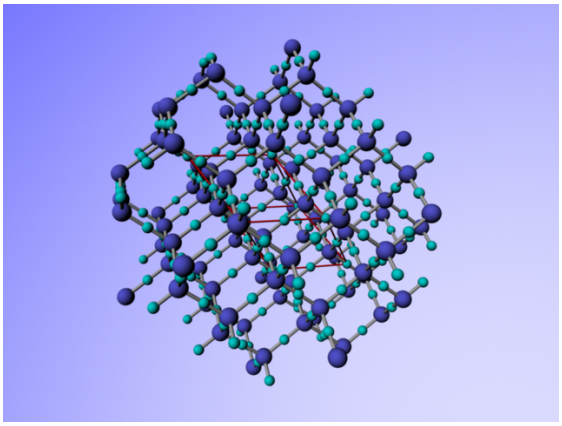

図 4.24: 高温型 cristobalite

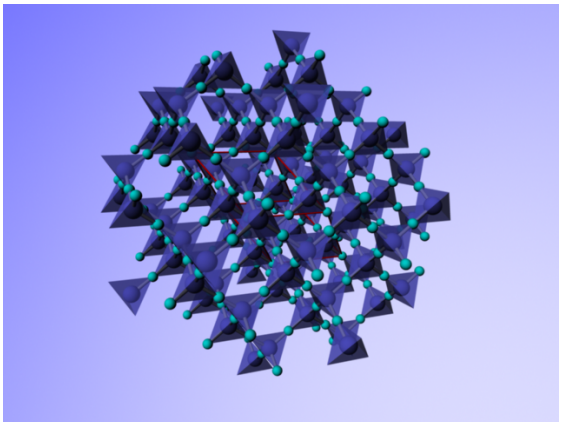

図 4.25: 高温型 cristobalite(四面体)

第5章　物理現象の視覚化

5.1　Fe-Cuの核生成のモデル

3DCGによるFe-Cuの核生成モデルについて解説する．

5.1.1　核生成モデルの作成

Fe-Cuの核生成モデルを表現するにあたって，web上のbcc構造のデータ[9]を入力ファイルとし，セルを作成した．これがFeのセルとなる．次にFeのセル上にスケール値を0.1大きくしたCu原子となるボールを配置した．さらにFeの配色を透明に，Cuの配色をblinn（小節3.2のcolorライブラリ参照）で表すことで，Fe中にCu原子が配置されているように見える．また，Cu-Cu結合の不安定性を表現するため，Mayaのパーティクル機能を利用した(図5.1)．

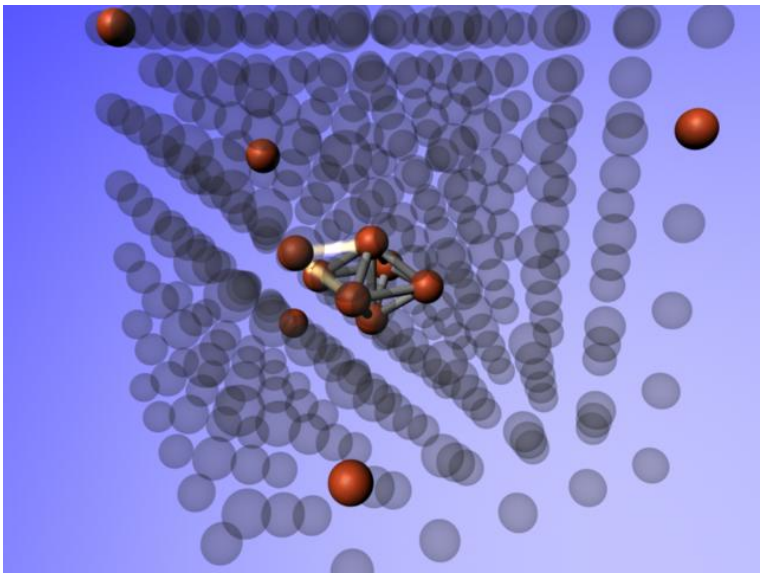

図 5.1: 3DCGによる核生成モデル．黒透明がFe原子．わずかに光を反射している茶色がCu原子．黄色いボンドがパーティクルを用いている[7]

5.1.2　核生成モデルの再現と検証

小節 1.2 で記したようにクラスターの状態は $f(n,t)$ で示される．クラスターは原子数と時間に依存するため，時系列順に核生成の様相を図 5.2 から図 5.11 に表した．

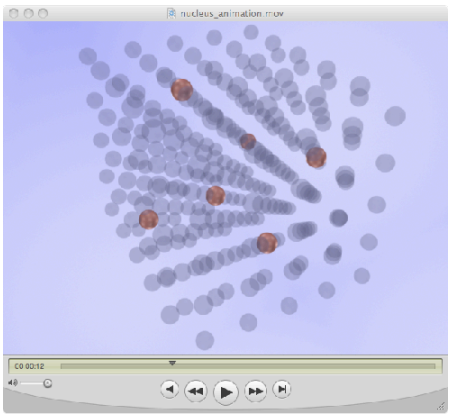

図 5.2: $f(1,t)$

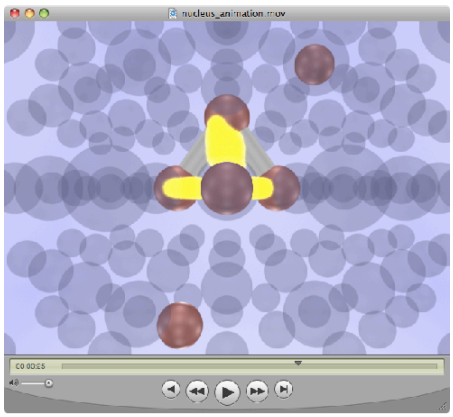

図 5.3: $f(3,t) \rightarrow f(4,t)$

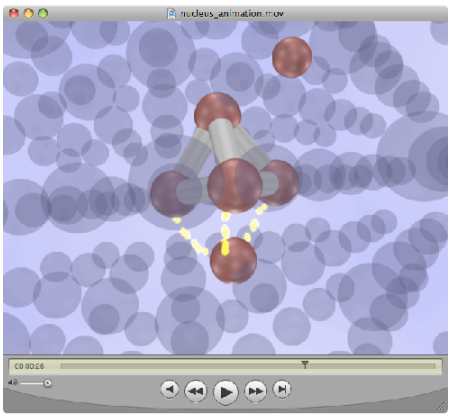

図 5.4: $f(4,t) \rightarrow f(5,t)$

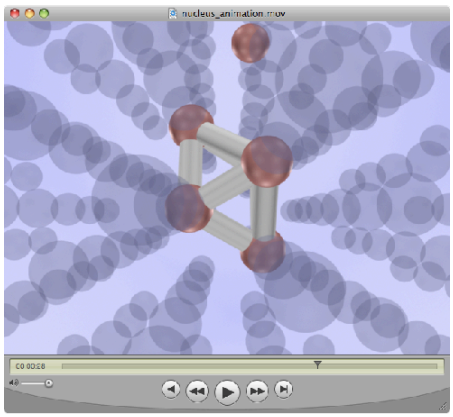

図 5.5: $f(5,t)$

初期状態では図 5.2 のように bcc 構造の Fe 中に Cu 原子がランダムに配置されている．Cu 原子は Fe 中を拡散しながら図 5.3，図 5.4，図 5.5 のように隣接するとクラスターを形成する．

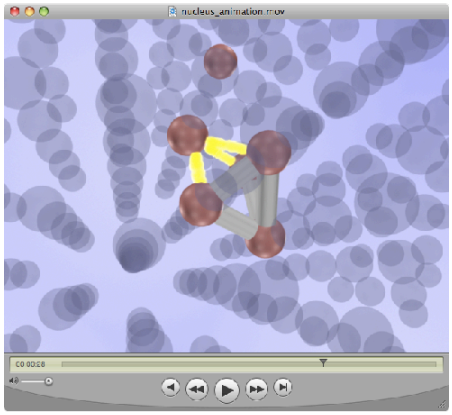

図 5.6: $f(5,t) \to f(4,t)$

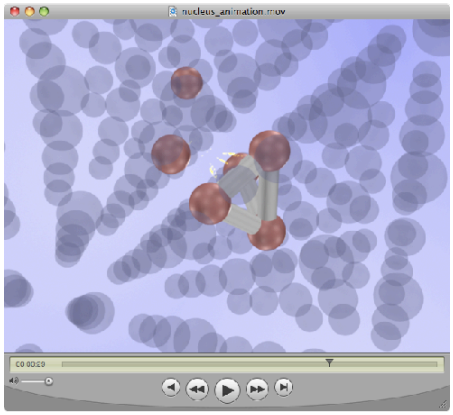

図 5.7: $f(5,t) \to f(4,t)$

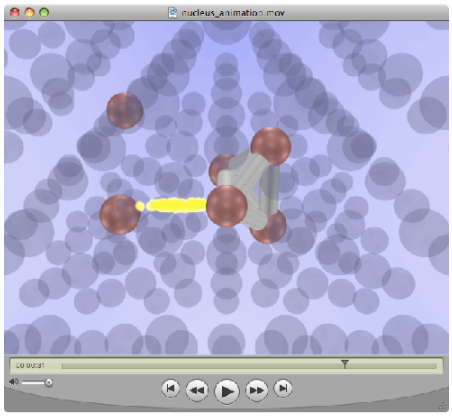

図 5.8: $f(5,t) \to f(4,t)$

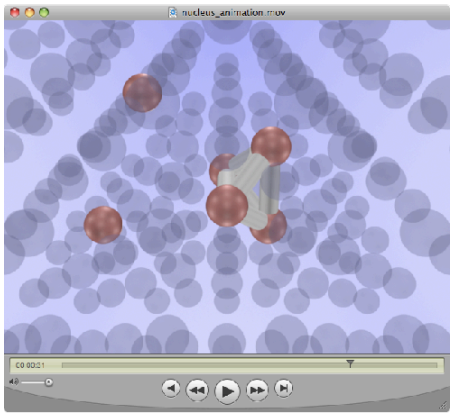

図 5.9: $f(4,t)$

クラスターは成長するだけ出なく，図 5.6 から図 5.9 のように縮小する場合もある．

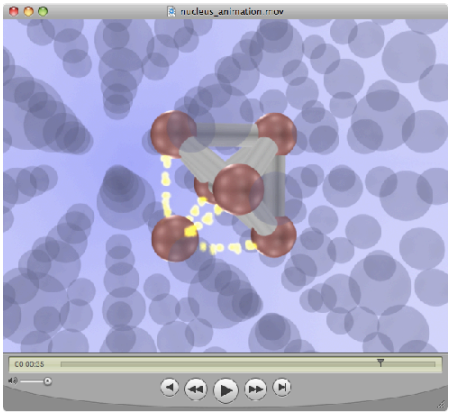

図 5.10: $f(5,t) \rightarrow f(6,t)$

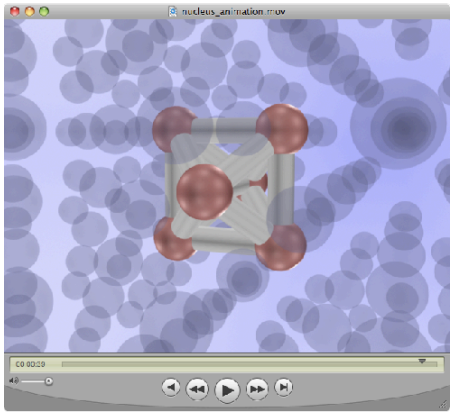

図 5.11: $f(6,t)$

図 5.2 から図 5.11 によって，クラスターが吸着と離脱を繰り返し成長していく様相を再現できた．
この Cu クラスターは Fe 原子に取り囲まれている．例えば $n = 4$ の場合には，線状あるいは面状の配置も考えられるが，Cu はできるだけ Fe との結合を避けるため，球状に集まる．n が大きくなったときにもこの傾向は見られるが，必ずしも表面積を下げるだけではない．bcc-Fe と bcc-Cu の界面エネルギーは {110} 面，{111} 面，{100} 面でそれぞれ 0.24，0.38，$0.60 J/m^2$ となっている．そのため，Cu クラスターは {110} 面で囲まれた構造を好む傾向がある．
3DCG で表すことによって，クラスターの状態変化を大きさだけでなく，構造を加味した視覚化を可能にした．

5.2　SiC 表面拡散の視覚化

SiC 表面拡散のモデルについて解説する．

5.2.1　SiC 表面拡散のモデルの作成

SiC 表面拡散を表現するにあたって，モデルは 4H-SiC の {0001} 面となる．西谷研究員によって第一原理計算によるシミュレーション [2] が行われているため，POSCAR ファイルが存在する．その POSCAR と計算結果を入力ファイルとし，セルと拡散経路を再現したモデルを作成した．計算は 4H-SiC の Si 面，C 面にそれぞれ 3 ポイントずつとり，エネルギー計算を行っている．エネルギーの高さに色で変化をつけることで表面上のエネルギー分布を表した．C 原子はエネルギーの低い場所を通るのでそこが拡散経路となる．

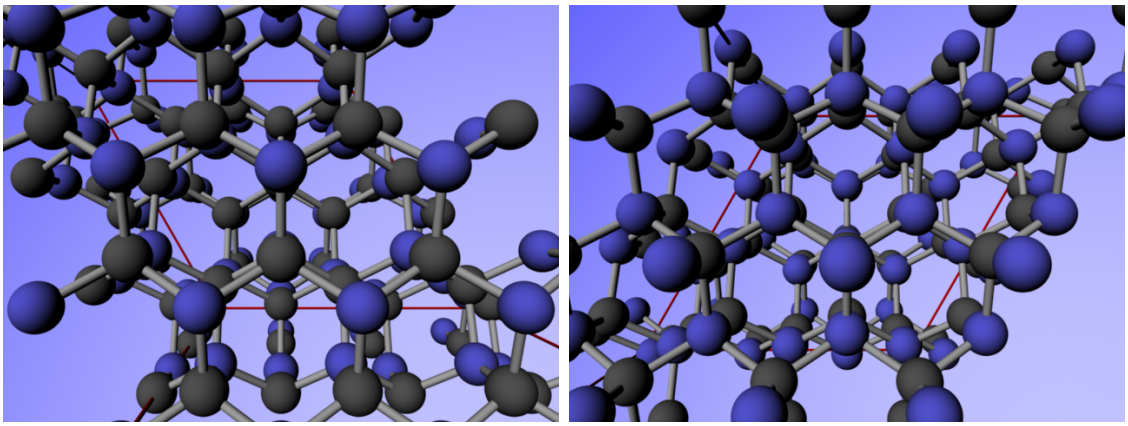

図 5.12: 図は両方とも 4H-SiC の {0001} 面となる．青が Si 原子，黒が C 原子となる．しかし，一番手前の Si 面から C 面までの距離が違う．このような構造の違いから，左を Si 面，右を C 面とする．

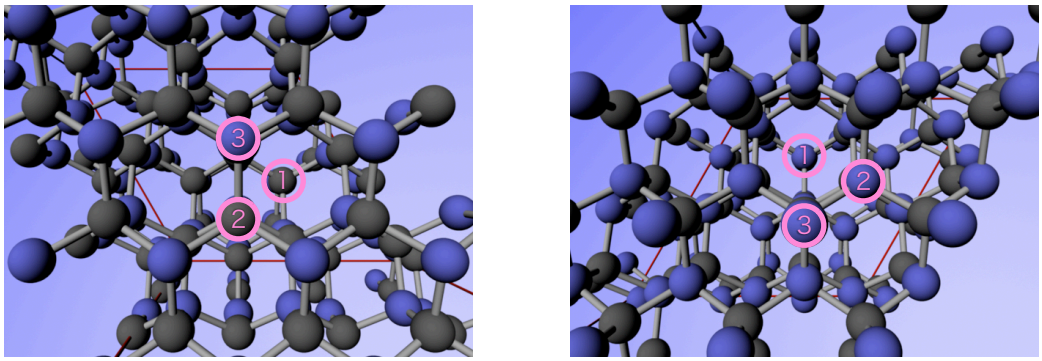

図 5.13: 左が 4H-SiC(0001) 面上への C 原子の配置場所，右が 4H-SiC(000-1) 面上への C 原子の配置場所.

表 5.1: Si 面のサイト別に付着させた C 原子のエネルギー

場所	(eV)
サイト 1	-7.85
サイト 2	-8.847
サイト 3	-5.806

表 5.2: C 面のサイト別に付着させた C 原子のエネルギー

場所	(eV)
サイト 1	-8.024
サイト 2	-8.028
サイト 3	-4.010

5.2.2 SiC 表面拡散の再現と検証

図 5.14 は，Si 面と C 面上における C 原子のエネルギー分布の模式的に表した図である．C 原子はエネルギーの低い場所を通ので，C 原子はエネルギーの低いサイト１とサイト２を通り拡散する．ここで注目すべきは C 面のサイト１とサイト２のエネルギー差が低いということである．これは，C 原子が C 面上では高速に拡散することを意味している．

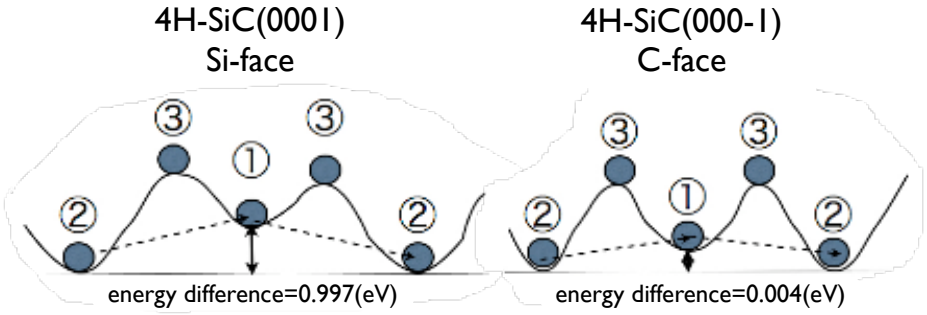

図 5.14: Si 面と C 面上における C 原子のエネルギー分布の模式図．C 原子はエネルギーの低い場所を通る．つまり，C 原子はエネルギーの低いサイト１とサイト２を通り拡散する．

図 5.15 と図 5.16 は，4H-SiC の {0001} 面上にサイトごとに C 原子を付着させたときのエネルギー分布を表している．赤い色がエネルギーが高く，青い色がエネルギーが低いことを示している．C 原子はエネルギーが低い場所を通るので青色に近い箇所を結んだ経路が C 原子の拡散経路となる．また，5.15 は経路上に青と深緑がある．これは経路上エネルギー差があることを示している．一方，5.16 には，経路上に色の変化が見られない．これはエネルギー差がほとんど無いことを示している．つまり，C 原子の拡散速度は Si 面上に比べ C 面上の方が高速であることを表している．

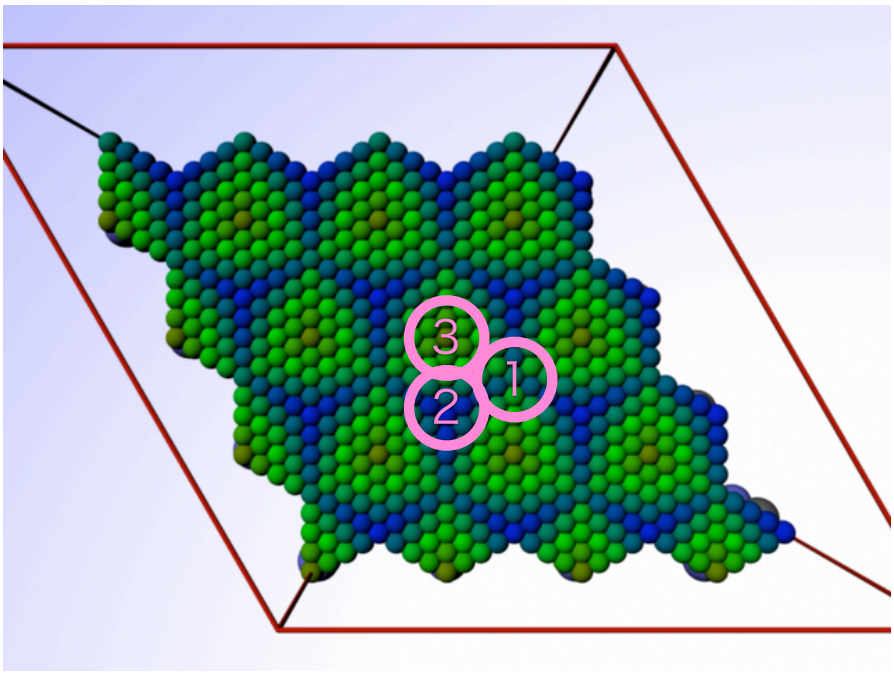

図 5.15: Si 面上における C 原子のエネルギー分布．C 原子はエネルギーの低い場所を通る．つまり，青色に近い箇所を結んだ経路が C 原子の拡散経路となる．エネルギー分布を面上で一望できることで，拡散経路が一目で解る．

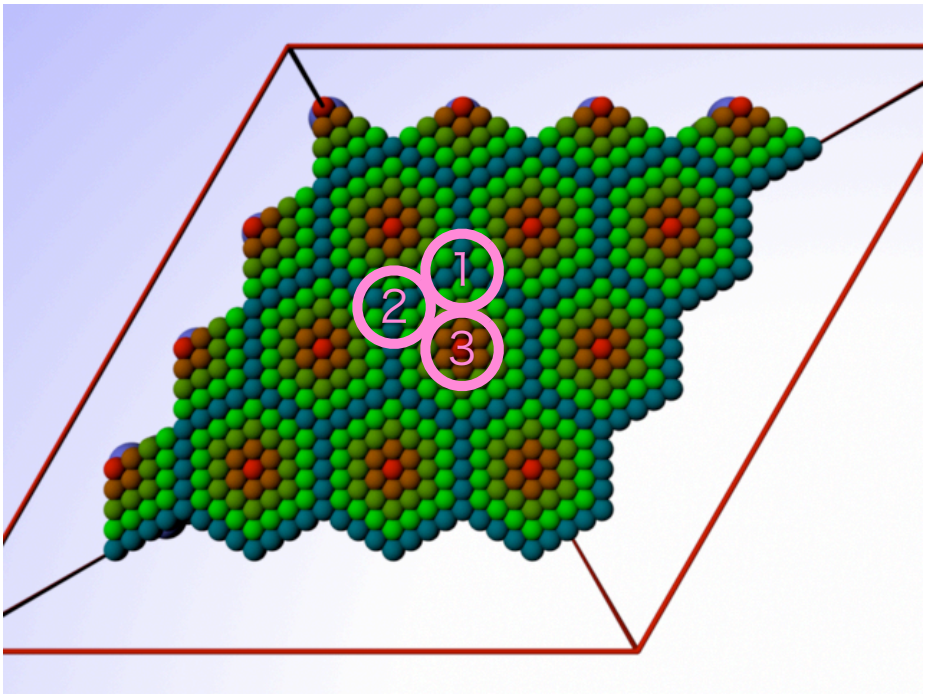

図 5.16: C 面上における C 原子のエネルギー分布．C 原子はエネルギーの低い場所を通る．つまり，青色に近い箇所を結んだ経路が C 原子の拡散経路となる．エネルギー分布を面上で一望できることで，拡散経路が一目で解る．

第6章　総括

[Ruby - MEL - Maya] のシステムの成果を以下に記す.

- Maya はグラフィックス性に優れ，自由度も高いソフトであるが，物理シミュレーションには向いていなかった．Ruby にて，線形計算などの数値計算を行い，その結果を MEL スクリプトに反映させることで解決できた．[Ruby - MEL - Maya] システムでは二種類のスクリプト言語を用いているにも関わらず，ライブラリの充実によって Ruby 言語のみでモデルの作成を行えるようになった.

- 格子モデルの視覚化については，事実上，'Crystal Lattice Structures' でが公開されている全ての結晶を表現できるようになった．加えて，第一原理計算ソフト VASP の入出力ファイルからの視覚化を実現させたことで，計算準備として対象モデルの構築が行えるようになり，構造緩和を行った計算後の構造を視覚的に解析できるようになった.

- Maya はアニメーションの制作が可能となっているため，時間変数を含む物理モデルの再現を可能にした．さらに，グラフィックス性を生かした多様な表現を用いることで，Maple などのソフトに比べて，より印象深いアニメーションとなった.

- 小節 5.2 では，エネルギー分布をセル上に色で表すことで拡散経路がはっきり見える．Maya によるエネルギー分布図は，模式図のそれに比べると解りやすくなり，研究の助けとなった．また，格子モデルやクラスターの様相と違い，ポテンシャルを表せたことで，本システムの柔軟性を示すモデルとなった.

参考文献

[1] 黒田登志雄　著,『結晶は生きている　その成長と形の変化のしくみ』,（サイエンス社　2006　初版第 10 刷発行）.

[2] 坂本憲　著,『SiC 単結晶成長の原子レベルシミュレーション』,（関西学院大学　理工学研究科　情報科学専攻　修士論文　2008）.

[3] 森口 晃治, 関 彰　著　「シリコンー酸素間の局所相互作用から見たバルク物性ー」　(住友金属　Vol.47 No.3 1995 P101).

[4] 西谷滋人　著,『固体物理の基礎』,（森北出版 2006）.

[5] 西谷滋人　著,『バルク中のナノサイズ熱力学と相安定性』,（まてりあ 第 46 巻 第 3 号 (2007) 創立 70 周年記念特集別刷　「材料科学の課題と展望?ナノマテリアル・環境材料を中心として?」　2007）.

[6] 阿部知弘　著,『MEL 教科書ー Maya プログラミング入門ー』,（株式会社ボーンデジタル　2004）.

[7] Dariush Derakhshani　著,『Maya 入門編』,（株式会社ボーンデジタル　2008）.

[8] 『Crystal lattice Lattice Structure』 http://cst-www.nrl.navy.mil/lattice/.

[9] 『Crystal lattice Lattice Structure：bcc 構造のデータ』 http://cst-www.nrl.navy.mil/lattice/struk.xmol/a2.pos.

謝辞

本研究を遂行するにあたり，終始多大なる有益なご指導，及び丁寧な助言を頂いた西谷滋人教授に深い感謝の意を表します．

また，本研究の進行に伴い，西谷研究員の皆様にも様々な知識の提供，ご協力を頂き，本研究を大成することができました．最後になりましたが，この場を借りて心から深く御礼申し上げます．